NATIONAL WEATHER SERVICE
OFFICE OF HYDROLOGIC DEVELOPMENT

Science Infusion Software Engineering Group (SISEPG)
Software Peer Review Guidelines

Version 1.7

Table of Contents

1. Introduction

The purpose of this document is to put forth guidelines for implementing software peer reviews within the Office of Hydrologic Development (OHD) at the National Weather Service. Peer reviews will be conducted as part of the software validation process and will be performed during the Hydrologic Operations and Service Improvement Process (HOSIP) Stage 4 (Operational Development) as well during software maintenance. It is highly recommended prototyping software undergo peer reviews as well, though they may not need to follow the guidelines discussed in section 4.0.

2. Peer Review Definition

A peer review is an assessment of a product conducted by a person or persons of similar expertise to the creator of the product. From a software development perspective, the product being assessed is source code. The persons performing the review are programmers, not including direct supervisors or managers. In the software world, the term peer review is often used interchangeably with the term code walkthrough. Strictly speaking, the two are not the same. A code walkthrough is the act of tracing the logic and validating the assumptions of a software module using test cases.

2.1 Advantages of Performing Peer Reviews

Performing software peer reviews has the following advantages:

- Most important, they promote software that is easy to read and maintain
- They serve as a mechanism for enforcing the OHD General Programming Standards.
- They serve as a mechanism for enforcing the OHD Language-Specific Programming Standards.
- They can catch bugs early.
- They are or will be required by our software delivery customers, that is, AWIPS and NEXRAD

The OHD General Programming Standards and OHD Language-Specific Programming Standards apply to all types of software tasks, small and large. The peer review is the mechanism which ensures that software developers are following these standards and guidelines when writing programs.

2.2 Disadvantages of Performing Peer Reviews

Performing software peer reviews has the following disadvantages:

- They can require resources that are needed for other projects
- They can be mired down in disputes over personal programming styles and preferences
- They often require follow up action to make sure that the software has been modified in accordance with peer suggestions

Peer reviewers should strive mightily to evaluate the product, NOT the person.

3. Peer Review Checklist

The peer reviewer(s) shall use electronic versions of the General Programming Standards and Guidelines Peer Review Checklist as well as the language-specific peer review checklists when performing a peer review. The reviewer may use the reviewer's comments section to add additional comments or suggestions. The checklists can be found at

http://www.nws.noaa.gov/oh/hrl/developers.html

4. Peer Review Guidelines

Peer Reviews shall either be in the form of an informal/ad-hoc peer review or a formal peer review. Which is to be performed shall be decided by the project leader, with the approval of the Project Area Leader or Group Leader.

The following sections are to used as guidance to help the project leader to make a decision on which peer review should be performed.

4.1 Scope of Software Development Tasks

It is difficult to develop a metric which precisely defines the scope of a software project and how detailed of a peer review is required. The number of lines of code written or modified is the metric chosen here, but this is subjective. As an example, a maintenance task may involve several hundred lines of code but still may qualify for an informal peer review. For the purposes of this document, software tasks have been divided into three categories: Small Maintenance Tasks, Application Enhancements/Significant Maintenance Tasks and New Applications/Large Software Projects.

4.1.1 Small Maintenance Tasks

Small maintenance tasks are those undertaken to fix or slightly enhance existing software functionality. Small maintenance tasks typically affect 1 to 100 lines of code in an existing program module. An informal/ad hoc one-on-one code peer review is appropriate in this case. Examples of small maintenance tasks include:

- Changing the appearance of a graphical user interface
- Fixing a null pointer read which is resulting in an application crash
- Modifying a program to use a binary search for locating stream gage IDs, replacing an already existing brute-force linear search.

Pair Programming is another way to conduct informal peer reviews. In this technique, software developers work side-by-side reviewing the code as it is developed.

4.1.2 Application Enhancements/Significant Maintenance Tasks

Application enhancements and bug fixes which require the addition/modification of 100-500 lines of code fall into this category. An informal/ad hoc peer review is still appropriate, although it is better if two or more peers evaluate the software. Examples of application enhancements and significant maintenance tasks include:

- Adding a new graphical user interface to an existing application
- A subroutine to decode a new HADS data format

4.1.3 New Applications/Large Software Projects

New Applications and large software projects which involve the addition/modification of over 500 lines of code fall into this category. The peer review should be formal, requiring that several peers are given adequate time to examine the code. The meeting of the peers and the developer is more structured.

5. Informal/Ad hoc Peer Reviews

Once a small maintenance task or application enhancement is completed, the developer shall request a coworker or two to perform a peer review. The developer should review the changes made with the peer reviewer(s). The developer should make the peer reviewer or reviewers understand:

- The nature of the maintenance task or enhancement
- The desired behavior and output of the modified software module
- The changes made to implement the desired behavior

The peer reviewer is responsible for completing the peer review checklists and returning to the developer. The developer will correct any issues. The developer will then review the source code again with the same reviewer(s). When the peer review is complete and the issues resolved, the developer will inform the project leader that a peer review was performed and give the project leader their completed peer review checklists. In the future, the peer review checklists may be stored in HOSIP DOORS database.

For application enhancements and larger maintenance tasks, the developer must include input from at least two peers.

6. Formal Peer Reviews

It is not practical to review every line of source code in such projects requiring a formal peer review. Instead, the developer and project leader shall select a subset of routines for review. The developer will then select a group of 2 or 3 other developers with subject area knowledge and provide them with electronic copies of the routines. The reviewers will be given at least 5 business days time to review the code. The developer will schedule a peer review meeting. Prior to the meeting, the reviewers should review and complete the peer review checklists. During the meeting, the developer will discuss each of the software modules asked to be reviewed. The reviewers will report any problems

they found. The developer is then responsible for addressing the problems found during the code review.

When the peer review is completed, the developer will inform the project leader that a peer review was performed and give the project leader the completed peer review checklists. In the future, the peer review checklists may be stored in HOSIP DOORS database.

7. Reviewer Comments/Action Items

The comments and action items placed forth by the reviewers shall be classified into three categories:

- Cosmetic Items– The reviewer is pointing out items related to internal documentation, control block indentation and variable naming conventions which do not conform to the OHD General Programming Standards or the OHD Programming Language-Specific Standards
- Minor Items – The reviewer is pointing out items related to the algorithmic efficiency of the software. The software will still work, but it will not work as efficiently as it potentially could. These may also be items which do not conform to the OHD General Programming Standards or the OHD Programming Language-Specific Standards.
- Severe Items – The reviewer is pointing out items which will cause the program to not work correctly or crash.

Each comment must be placed into one of the above three categories. In addition, each comment must have a detailed statement indicating the problem and a suggested corrective action.